A2-Central™

formerly Open-Apple

April 1992
Vol.8, No.3

A journal and exchange of Apple II discoveries

ISSN 0885-4017
newstand price: $3.00
photocopy charge per page: $0.20

Programming is Simple

By Jeff Smith

There's nothing magic about writing computer programs. In *HyperStudio*, we call programs "scripts." And "scripting" is simply the act of getting the computer to do what you tell it to. You think up a series of actions, such as showing the numbers 1-10 one at a time, then you enter some lines of text into the Editor to make it happen.

Simple Scripting

The first lesson of Scripting is that scripts are nothing but a series of requests to a computer. A list, we might call it, of what you want to happen, in exactly the order you want it to happen.

Let's try something. Go into *HyperStudio 3.0*. Make a new stack, and create a button named "Example 1". When you get to the 'Button Actions' screen, you'll notice there is an option called 'Scripting language.' Select it.

Now you're looking at the SimpleScript editor. If you knew the language the way I do, you could, at this point, just type in some command lines and get any result you wanted.

Let's try it. If I remember right, we were talking about a program that would let you show the numbers from 1 to 10. To have it display the number 1, you type in the line 'Show message "1"'. Then, after hitting Return, you type 'Show message "2"'. And so on, until all 10 are typed.

Just for your information, you don't need to actually type all ten lines. Just like your favorite word processors, the SimpleScript editor supports the standard Apple IIgs cut and paste actions. So, if you select an already typed line, which triple clicking will do, you can press Apple-C. This makes a copy of the selected area. Move the cursor to the next line and paste (Apple-V) — you've just let the editor type in 90 per cent of the line for you. All you need to change is the number within the quotes and you're ready to move to the next line and paste again.

Also notice that you don't need to type in the first line of the following listing. The editor types the remark for you based on the name you gave the button when you created it.)

```
-- Example 1
Show Message "1"
Show Message "2"
Show Message "3"
Show Message "4"
Show Message "5"
Show Message "6"
Show Message "7"
Show Message "8"
Show Message "9"
Show Message "10"
```

When you finish with all ten lines, check over your typing — it should look like the listing shown here. To tell *HyperStudio* about this script, choose Save & quit editor from the File menu.

Now we are back at the old familiar *HyperStudio* 'Button Actions' screen. Since this is all that we want this button to do, finish the button creation process by clicking on the 'Done' button.

Hello! There is our white screen and new button. Before we try it, do a quick Apple-S to save our masterpiece of a stack to the disk. Then try your script by clicking the 'Example 1' button.

The moment you click, SimpleScript looks at each line, starting with the very first, and checks to see if it has any idea what you're talking about. If so, it performs the action. Since it knows 'Show message', you should see 10 dialogs appear, each with a different number. Each dialog waits for you to hit return or click on the Ok button before the program continues.

(If you mistyped something, SimpleScript will let you know by taking you back to the editor and highlighting the incorrect line. Fix it, choose Save & quit editor, and try again.)

As easily as that, you now know the first lesson of Scripting, which is summarized by the word, "Order". A language looks at and performs each line in a specific order— starting at the first and ending at the last. It performs all that magic one step at a time.

The second lesson of Scripting is just as easy. Suppose we wanted to let users type in their names, and we had big plans to display them with 'Show message'. Sure, we can tell SimpleScript to put up a dialog and let someone type in their name. That's easy. That's just the 'Ask Question' command. But how will we tell the computer to remember what was typed? How can we ensure that SimpleScript uses exactly the same result with the Show Message command?

Variables are the magic here. When SimpleScript sees a word it doesn't understand, it figures the word represents a variable that you want to create. Variables are little spots in memory that are named by the word you used and that SimpleScript uses to store data you associate with that word. You could also picture them as little labeled mail-

FOR ADDITIONAL SOFTWARE SUPPORT, DIAL "9", "POUND" THE EXTENSION NUMBER DIVIDED BY YOUR ACCOUNT NUMBER, HIT "STAR", YOUR DOG, AND WHOEVER ELSE IS AROUND, BLOW INTO THE RECEIVER TWICE, PUNCH IN YOUR HAT SIZE, PUNCH OUT YOUR LANDLORD, DO THE CARIOCA, ...

boxes waiting for their mail.

Shall we try it?

Create a second button, name it 'Example 2', and go to the SimpleScript Editor through the 'Button Actions' dialog again. Then type in the following script—

Remember that you don't need to type in the first line.

```
-- Example 2
Ask question "What is your name?" with MyName
Show message MyName
```

Most of that first line makes sense. It's going to put up a dialog that asks a question based on that text and then wait for the user to type in something. But wait. What is that 'with MyName' business?

SimpleScript requires that 'with MyName' syntax because it needs to know the name of the variable you want to put the result (whatever is typed in) into.

Notice there is something different about the second line — different from the first time we used that command. "MyName" doesn't have quotes around it like the numbers in Example 1 did. No quotes tells SimpleScript that "MyName" is a variable, and to use the *contents* of that variable, while quotes tell SimpleScript to display the literal text between the quotes.

Go ahead and choose Apple-S to save Example 2, choose Done from the Button Action menu, and do another Apple-S to update the stack on the disk.

Now you can try it. Notice that whatever you type in is remembered, and, when you hit return or click on the OK button, displayed by the Show Message line.

So variables are definitely the second lesson of Scripting. Think of them as little boxes in the computer somewhere with names like 'It' or 'MyName' that you can put values into and retrieve later with SimpleScript commands.

Okay. You're doing very well. Now you are ready for lesson three. Before we get into it, here's another SimpleScript secret. If you've got a button that already has a script, a short cut to entering the editor is to hold down the Apple Key and click on the button. This calls up the Debugger, the features of which we will explain later in this article. But one of the buttons the Debugger offers is "Editor." Guess where that takes you? Go ahead and use this short cut with your Example 1 button.

See how long your listing is? And how similar each line is? Wouldn't it be neat if there was some way to just tell the language to show the dialog ten times, and only have one 'Show message' line? There is.

Lesson three is that looping is the backbone of Scripting. Watch what happens when we change this script.

Choose New script from the File menu to erase the current script, and type in 'Example 1 rewritten' when it prompts you for a remark. Then type in each of the following lines except the first. Don't worry about indenting the 'Show message' line— SimpleScript will take care of that for you.

```
-- Example 1 rewritten

For MyCount = 1 to 10
 Show Message MyCount
Next
```

Much easier to type this time, isn't it? But what does this all mean? The first line 'For MyCount = 1 to 10' tells the language that we want to use 'MyCount' as a variable to count with. We want to start 'MyCount' at one and keep repeating until 'MyCount' equals ten.

Okay. That sort of makes sense. 'MyCount' starts at a value of one. But how is 'MyCount' changed? What adds one to 'MyCount'? 'Next' is the culprit. But we'll get to that in a second...

Line 2 looks the same as in Example 2. Again we are using the value from the variable (1 the first time through) instead of raw data.

But the third line is totally new. 'Next' tells SimpleScript that this is the end of the loop. Now SimpleScript has to decide whether it is time to go back up and repeat the routine or to let it fall through to whatever command comes after 'Next,' if any. But that won't happen until 'MyCount' is ten, remember?

'Next' has a second function too. It adds one to 'MyCount' and checks to see if we've hit the maximum from the 'For' line. So the first time through, it pumps 'MyCount' (which is one) to two. Then it looks and decides that two is less than ten, so it loops back up to the Show message line. Only when 'MyCount' is ten or more will it execute a line after the 'Next'.

Press Apple S to exit the editor, and try it. Looks just the same as before, doesn't it? But thanks to the looping capabilities of Scripting, it's much easier to type and understand.

Now you know the basic three lessons of Scripting: line order, variables, and looping. Try to visualize some simple script of your own — like displaying only the numbers from 7 to 10, or maybe numbers up to 100. The more you fiddle, the more you'll get a feeling for exactly how it all works.

And it's fiddling like this that teaches far more than formal lessons.

Simple Made Easy

Now lets look at some of the ease-of-use features of SimpleScript and *HyperStudio 3.0*.

Some people find it hard to believe, but programming your Apple IIgs can be simple — with HyperStudio 3.0 and SimpleScript. Despite the initial similarity to a normal Apple IIgs word processor, there is a curious mixture of ease-of-use and power apparent immediately.

SimpleScript has a number of helpful features that are directly responsible for it being called 'Simple'Script.

First, you never need to enter a title remark. When you create a new button with a script attached, SimpleScript finds out what you typed as the button's name and uses that as a title remark. (A remark, for those wondering or those wandering, is a line in the editor that SimpleScript will ignore. Since you don't want SimpleScript to actually execute script titles, author's names, or copyright notices, for example, then you would put the '—' in front of them.)

Okay, let's create a new button called 'Who are you?', position it where we want, and choose 'Scripting language...' from the Button Actions screen. Now we're looking at the editor, with that title '— Who are you?' already typed as a remark.

Let's start with something easy. Suppose we want to have a little dialog pop up with our name in it...

In a normal language, we would be daunted by all that white space. We wouldn't have the first idea where to look. We'd probably even have to wade through some thick reference book to even get an inkling what to type.

But not with SimpleScript. If you pull down the menus, you'll find commands listed. Go ahead and do it. Every command known to the language is accessible through this set of menus.

Let's try it. Pull down the Objects menu and you'll see something called 'Show message'. Choose it.

Now the language puts up a dialog box whose sole purpose in life is to help you build the line. All of the raw text you'd normally type is there (like 'Show message' and 'with'), but you need to fill in the holes. First, it wants to know what text you want to put in the dia log box. From the first pop-up menu, where it says — Select —, choose the '(type)' option and enter your name surrounded by quotes. Second, it wants to know which style of icon you want in the box. From the second pop-up (— Select —) pick any icon that sounds nice.

When you are finished, click 'Ok'. That 'Ok' inserts the line into the current script at the current position of the cursor. And, just like that, you've got your program.

To try it out, press Apple-S or choose Save & quit editor from the File menu. This gets you back to the Button Actions screen, where a 'Done' will finish the button creation process.

Since that 'Save' operation only changes the stack, not the file on the disk, you'll probably want to save your masterwork to the disk at this point.

Click on the button, and a dialog box comes up with an icon (if you chose one) with your name in it. It works!

See how simple that was?

A second example of power through simplicity appears when you come to a situation in the language where you need some information that you don't have. For example, suppose you are writing this neat stack about the Black Death and want the screen border on the Apple IIgs to be Black to give that properly moody look to it.

The problem is that the colors on the Apple IIgs are numbered 1-16. They don't have nice easy names like 'Black'. So how do you know what number to enter for Black?

If you go back to the editor—

Wait. There's another way to get to the editor if you don't like the debugger trick I showed you earlier. Hold down the Shift key and press Tab to switch HyperStudio to edit mode, which turns the cursor into a pointer (you could also do this by selecting the pointer cursor from the Tools menu). Now hold down the Apple key and double click on the button. The Button Actions screen pops up, with the 'Scripting language...' option checked. Click there twice to re-enter the editor and see your script.

Okay. Whew. Now, regardless of how you got there, you are back at the editor. We want the screen border to darken to black after it prints our name. Since we learned from the example above that SimpleScript inserts the line where the cursor is, you better move the cursor down to the end.

Great. Now, from the Screen Menu, choose the option 'Set border color'. This dialog is only asking for one bit of help from you— the number of the color Black.

Do you know it? No? Well, don't feel bad. I don't, either. And that's why there's a '(which?)' option. When you see that option, it means SimpleScript is willing to help out by showing you all the choices and letting you choose the one you like.

Set border's (which?) option brings up a dialog with a button in it. By clicking this button, you can change the current border color of your Apple IIgs. Just keep clicking on the button until you see the color you like — black for Black Death if you're still following our strange storyline. Then choose OK, and it types the number for the color you chose into the pop-up for you.

So, not only does SimpleScript help you along by entering 90 per cent of the line for you, in many places it actually tries to help you with the other 10 per cent.

Click on the Ok to actually insert the line and then press Apple-S to Save & quit the editor.

Try it. Again, an easy success.

Okay. Let's try something a little tougher. Let's play with reality a little. Let's try to get some idea of what is actually happening deep inside a script.

Besides giving you a headache, this will also give you a taste of the next extremely helpful aid in SimpleScript— the debugger.

Begin by creating another new button— 'Debug test'. Now that you've kind of got a feel for how things work, I'm not going to lead you through every step anymore.

When you get to the editor, use the pull-down menus to enter the following lines:

```
-- Debug test
For It = 3 to 5
 Set border color to It
Next
```

First, choose the 'For' line from the Structure menu. It needs three bits of information from you. The first box is the variable you want to use (choose 'It'). The second is the starting value for the count (choose (type) and enter 3 with no quotes). The third box represents the value to end counting on (choose (type) again and enter 5 with no quotes).

When you click 'Ok', and SimpleScript inserts the line, you'll see a couple of new neat things. Notice that SimpleScript helps you along here, too, by automatically inserting a Next line in your script and automatically placing the cursor on the 'Next' line. This is to ensure that whatever line you enter next will be placed inside the loop (dare I say it again?) automatically!

The Set border color, just like before, is in the Screen Menu, and you need to choose the variable 'It' from the Pop-Up menu. Notice, though, that yet another neat thing about SimpleScript appears— the line automatically indents itself!

Okay. The moment you've got that script entered so that it looks exactly like the example here, go ahead and press Apple-S to save and quit.

Before you click the button, press Apple-S to save the stack to disk. When you do click the button, you'll notice that the border color changes, but it happens so fast it's hard to tell if the script actually changes the border color three times like it's supposed to.

How can we check a script to make sure it is doing exactly what we want? Enter the Debugger and take a look. (Remember how? Just press the Apple key while clicking on the button.)

The Debugger window has several features that are obvious. As I mentioned before, there is a button to take you to the editor marked 'Editor'. There are also three other buttons— Next (to tell SimpleScript to perform the line it is displaying), Cont (to 'Continue', telling SimpleScript to put away the debugger and go back to executing the script), and End (which tells the language to stop execution of the script entirely and return to HyperStudio.)

Notice that the title remark of the button is visible. That is SimpleScript showing you the line it is about to execute. Click on the Next button. The title is gone. Since remark lines don't do anything, SimpleScript has just moved to the blank line following the remark. Keep pressing Next until you see the 'For' command.

What does this 'For' do? This begins a simple loop. Loops need to know three things in order to work— where to start, where to end, and what variable to mangle to keep track of it all. In this line, the 'It' serves that variable function, while the first number (3) is the number 'It' will start at.

Press Next. Now you are looking at the line that sets the border color. What value is it going to use? The For line supposedly set the variable It to 3, but how can we check that? Is it even possible?

Of course. See the pop up control '— Select —'? Click on it and choose 'It'. This control lists every variable in the source listing. "It" shows a value of 3. So, the For line has started 'It' off correctly. The program seems to be working fine.

Press Next. See the border change to purple— color 3. Notice that the new line coming into view is the 'Next' command.

What does the 'Next' command do? It is the closing part of the loop. It, simply enough, adds 1 to the variable in question (It) and then checks to see if the result is equal to the ending value the 'For' specified (5). If it is not, then Next sends the flow of control back up to the line right after the 'For' line— back to our Set border color again.

Try it. Hit Next, and notice that It increases. The new line in the debugger is now the set color line again. Hit Next, and the border changes to dark green (4). Hit Next again, and "It" increments to 5. Hit Next again, and, once more there is a new color.

But now "It" is 5. Isn't that where we wanted it all to end? Yes. When you hit Next this time, the script ends.

It worked!

With a tool like the debugger, you can step through any script and see exactly, step by step, what it is doing. The SimpleScript debugger, as well as its other helpful features are all there to make programming the Apple IIgs just plain fun.

Secret HyperStudio tip #234. If the contents of a variable are too big to show in the gray box, the debugger will show the first twenty characters and a '...'. If you want to see the rest, just click there. This brings up a window with a scrollable text field. The entire contents of the variable will be visible. You can even change a variable's contents on the fly this way.

Science fiction theatre

by Art Coughlin

(Editor's Note: Back in January we promised to celebrate the 15th Anniversary of the Apple II by bringing you articles about people using Apple IIs "to create, to produce, and to simply have fun." Here's a good example of what we have in mind.)

Before I bought my Apple II Plus in 1982, I thought long and hard about what I was going to do with this "thing". It lacked the power and resources of the mainframes and minis that I was used to; typing letters, keeping track of finances and playing games just wasn't enough. I made up my mind that if I bought this machine, it was going to do things for me. It was going to make my life easier. That's what has driven me to the bleeding edge.

I began writing a menu driven house manager program (Master Control) in Applesoft Basic about 6 months after purchasing the Apple II Plus. I didn't set out to write a house manager, I was just 'testing' various features of the machine. Once I discovered the game port, I was hooked. I could read from the 3 'pushbutton' inputs and control 4 relays from the 'annunciator' outputs. The machine now had the ability to reach out beyond itself: it could detect a change in its environment and respond. **Wow!** It didn't take long to start filling those slots to extend the Apple's reach even further.

In 1987 I re-wrote Master Control in Pascal as an Apple IIgs Desktop application. Since then, the beast has grown to 3800 lines of code (and the experiment continues). So, what does all that code do? Well, Here's a rundown on the current features of Master Control (subject to change on a whim):

The Telephone Interface. Master Control acts like an answering machine. It answers the phone with my voice, records messages, and maintains a phone log with time and date stamps. I can dial in using a touchtone phone, enter a password, and get the house status (notification of various events, such as: mail arrival, basement flooding, fire, breach in security, number of phone calls received, and so on.). I can also use the touchtones to control devices in the house. If I get to work and remember that I left the kitchen light on, I can dial in and turn it off. I'm considering adding a feature that would allow friends to enter a password that would allow each of them to check for a message when I'm not home.

Master Control also has the ability to make a phone call. I can select a number and time to call and digitize a message for it to deliver. This comes in real handy if I'm sick at night and I don't want to wake up early in the morning to call in to work. I just set up a message, supply a phone number and tell Master Control when to deliver it. This feature can also be used for more normal things, like dialing a number to report an emergency (break-in, fire, flooded basement, etc.).

Clock Functions. I hate waking up to a clock ringing or buzzing, so Master Control calls me (via a speaker in the headboard) at the appointed time in the morning. If I don't get up by the third call, it turns on the bedroom lights. Not only does Master Control display the current date and time, but it also has the ability to chime or announce the time on the hour.

Care and Feeding of the Environment. I've hooked up a variety of sensors to the Apple IIgs. I have magnetic sensors on the doors and windows and a couple of infrared sensors in the house that can be activated at the click of the mouse. If any of these sensors detects a break-in, specific lights are activated, a note is posted on the Desktop and the computer begins to taunt the intruder. Of course, I'm not above having a little fun with the infrared sensors, such as having the Apple IIgs make rude remarks to passing guests. (A sick mind and a versatile computer make for some warped fun.) I've also put infrared sensors at the front door and driveway. When someone is detected, Master Control announces their arrival and activates a light and security camera in the driveway. The security camera is hooked to a VCR and can be viewed on the TV in the playroom.

I've installed some optical sensors in the cellar because I'm always leaving the lights on down there and forgetting to turn them off (for days!). Now when I turn on the lights, a note appears on the Desktop (and disappears when the light is turned off). Speaking of the cellar, it occasionally floods if the sump pump quits during a heavy rain, so I positioned a water sensor at the lowest point and connected it to the Apple IIgs. Now if the cellar floods, a note is posted on the Desktop and Master Control announces that "there is water in the basement".

I've even installed three temperature sensors: one for outdoors, one for the house interior and one to measure the Apple's internal temperature. Each of the temperatures are displayed on the Desktop along with the high and low exterior temperature for the day. I'm considering having Master Control announce the temperatures when it wakes me and adding them to the house status report over the phone.

Using a keypad out of a dead telephone and some heavy duty solenoids as door locks, I've managed to do away with key fumbling. To unlock a door, I just enter my entry code. Master Control will activate the solenoid, unlocking the door. When leaving, pressing zero on the keypad will lock all the doors. So when I drive up the driveway, a light comes on, I walk to the door, punch in my code and I'm in (and the doors lock behind me).

Master Control has the ability to control nearly any device, both in immediate and timer modes, that is plugged into the 110 volt house wiring. If the device is a light, brightness can also be controlled. In addition to lights, I have control over the furnace, air conditioner, fans, stereo, waterbed heater, door locks, external modem, infrared sensors and the Apple IIgs itself. Yes, that's right, the machine has the ability to set a timer event for itself, then power down and come back on later. The Apple IIgs can make use of this feature if it determines that it is overheating ("hasn't happened yet" he says crossing his fingers) or if it determines that the SPS (Standby Power Supply) has switched to the backup battery. During a power outage, it will set a timer event for itself for one hour and then shut down. When it powers back up, Master Control checks to see if the SPS is still on battery power, and if so, it repeats the cycle. The same is true for overheating, although Master Control has the option to activate fans and/or the air conditioner before deciding to shut down.

One reason I bought an Apple II was the flexibility it provided. Eight slots and the game port provided ample room for expansion and experimentation. This is particularly true of the Apple IIgs. I originally thought that it was "slot poor" since slots 3 & 4 were dedicated to internal functions and the other slots could only be used at the expense of the internal functions. However, hardware developers liberated slots 3 & 4 with stereo boards and accelerators and, with a little programming and careful planning, I can switch between internal and external functions for slots 1 & 2 by diddling with the Slot register.

****** Warning ******

Diddling with the slot register can be hazardous to your operating

system! Or, as Apple puts it in the Hardware Reference: "You are encouraged not to manipulate the Slot register bits under software control; you run a great risk of crashing the operating system."

Well, anyway, I've been getting away with it for over a year. Back to those slots; I wish I had more—my Apple IIgs is stuffed! Maybe I'll have to buy a second one so I can play with things like a *Video Overlay card* and *ComputerEyes GS*. Anyway, here's a rundown of what's plugged into my machine and why:

Slot 1 contains an Input/Output board with eight input and eight output switches. Serial port 1 is connected to a 4-way switch box with an Imagewriter II hooked to channel 1 and an X10 PowerHouse Computer Interface hooked to channel 2 (channels 3 & 4 are for future expansion). The X10 Powerhouse is a device that communicates using your electrical wiring. You plug a device into an X10 module and set the house code and device code on the module. You access each module using the specified codes. The X10 will support a maximum of 256 devices. There are a variety of modules you can buy, from stand alone modules that you can plug lamps and appliances into, to replacements for your current wall outlets and wall switches. The X10 also gives you the ability to set up timer events. You specify the device, the action (on/off), and the time the event should occur. If you are controlling a light, you also have a dimmer option. Another neat feature of the X10 is its ability to store timer events internally. Once the event is stored, the Apple doesn't even have to be on — the X10 does the work.

Slot 2 holds an old AppleCat modem, which is used as the voice communications interface to the phone line. The AppleCat controls a tape recorder for recording phone messages and provides access to phone I/O signals. It also has the ability to decode touchtones received over the phone line. Of course it's obsolete as far as data communications go, but you can't have everything. Serial port #2 is connected to a 2-way switchbox with a 2400 bps modem connected to channel 1 and another computer connected to channel 2 (more about that later).

Slot 3 contains a 10Mhz TransWarpGS with a 32k cache. At this speed, I run faster than a 16Mhz 386SX with Windows 3.0. To be fair, the 386 is fast enough; it's Windows that slows it down. (What a combination, a poorly implemented Graphical Interface on a makeshift DOS.)

Slot 4 holds a Supersonic Stereo board with sound digitizer. Actually, slot 4 is partially blocked, so I built a "slot extender" that allows the board to sit upside down between slot 7 and the case. I couldn't extend the slot outside the case because the back of the machine is already a mass of wires and the board has a short cable that plugs into the sound connector on the motherboard. This arrangement is dangerous and I had to take precautions to insulate the board from the case, motherboard, and *MemorySaver* board. The stereo output is routed to a Yamaha amp for some truely awesome sound. I love being able to digitize sounds and make use of them in various programs. Master Control talks. I wrote a clock NDA that announces the time (or chimes) and the database system I'm working on allows sound annotation for each record.

Slot 5 is completely blocked. I have two 3.5 drives and one 5.25 drive connected to the SmartPort. No room for a SuperDrive (sigh).

Slot 6 contains an Analog to Digital board that is used to measure things like temperatures and brainwaves. Oh, did I forget to tell you about the brainwaves? Well, read on.

Slot 7 contains an Apple HS SCSI card with two Chinook drives attached (20 & 50 MB). Future expansion potential here.

The memory slot has a MemorySaver with an 8 MB *Octoram* memory board (4 MB RomDisk and 4 MB free ram). I boot off the RomDisk, and even with 18 desk accessories (330 k), 41 fonts and a couple of extra cdevs and inits, it takes only 13 seconds to go from the beginning of the Welcome screen to the Finder's fully drawn menu bar.

The 9-pin game port on the back of the Apple IIgs is connected to a switchbox. I have a joystick and a *Thunderscan Digitizer* connected to that. This was an absolute necessity since I can't even get near the port through the mass of wires. The 16-pin game port inside has been extended out the back of the case and is used as an I/O device.

I've even made use of the video and sound ports on the back of the computer. The "headphone jack" is connected to a $6 stereo amp. The output from the amp is routed through a series of relays controlled by the 4 outputs on the game port. By toggling different relays on or off, I can route the sound to speakers in different rooms of my house. Current speaker locations are the bedroom (so the Apple IIgs can wake me up in the morning), the front door (to greet visitors), the computer room (of course), the modem (so the Apple IIgs can talk on the phone) and downstairs in the playroom (so I know what's going on upstairs). The video port output is hooked into one of the external input jacks on my TV in the playroom. At the push of a button on my remote control, I can switch channels and monitor the Apple IIgs upstairs. This comes in real handy when I'm doing a long download from GEnie. I don't have to sit there and watch the blocks go by, I can go downstairs and read or watch TV while I monitor the progress. I've also routed the phone sound from the AppleCat to the TV, so I can let the computer answer the phone, while I monitor the call from downstairs.

I like to page through electronics catalogs. Every once in a while I'll see something that triggers an idea. That's how I came to purchase the *Biosone II* bio-feedback machine. You strap on a headband with a couple of electrodes and you get feedback on your alpha or theta brainwaves via a little flashing light and/or beeping from a speaker. I was completely unimpressed with the flashing light and the beeping speaker. What sold me was the expansion port on the back that provided access to the electronic signals. I ran the signals into the Analog to Digital board in the Apple IIgs and wrote an NDA to scale and display the signals. Now I can activate the NDA while doing programming and watch my brainwaves zip by (yeah, I know I'm a sick pup). I've also been toying with the idea of trying to manipulate devices hooked into the computer. If I can maintain a certain level of alpha or theta wave activity for a specific amount of time, I could turn specified devices on and off (or something like that). Now if I could only program directly from REM sleep.

Another great source of ideas is plain old day to day living. Take, for example waiting for the mail to arrive. The mail box sits right outside my front door. In the winter, every time I open the door to check the mail, I get hit with a blast of cold air and since the mail doesn't arrive at the same time each day, I have to check several times. One day, after being hit with a particularly cold blast (or so it seemed, being in a bathrobe and hungover at the time), inspiration struck. I grabbed a magnetic switch and 50 ft of wire, glued the magnetic switch to the mailbox lid, ran a +5 volt line from the machine to the switch and a line back to 16 pin game port. I added a few lines of code to Master Control, digitized my voice, and now when the mailman deposits my mail, the Apple IIgs announces that "the mail has arrived" and posts a note on the Desktop.

And, of course, don't forget necessity. When I'm deep in the bowels of a programming project, I hate to be interrupted. Usually, the phone will ring, dragging me back to reality; only to hear some voice on the other end trying to sell me their brand of snake oil. My solution was to write an NDA that would wait for the phone to ring, answer on the first ring and record a message. Now I can ignore the phone and continue programming without missing a keystroke.

So, what's next? Well, I recently purchased a (gasp) 386sx notebook. Now, before you start forming the lynch mob, let me say that what I wanted was an Apple IIgs notebook. I don't get much programming done in the summer since I don't want to be stuck in the house. With a notebook, I can transfer source code to the notebook, put the notebook in a backpack and go cycling. I can relax on the banks of

the Delaware and get some work done. The other reason I got the notebook was I wanted to be able interact with the Master Control program without having to go upstairs to make selections. I've setup a menuing system on the notebook that communicates with the program on the Apple IIgs (at 19,200 bps). Since the "remote site" is the table next to my easy chair, I needed something compact. Besides, it gives me perverse pleasure to make the notebook subservient to the IIgs.

Has all this made my life easier? Well, yes and no. When everything is working right (which is **most** of the time), yes! But, Murphy's Law has a way of popping up and when it does, it can be Hell in TechnoLand—late nights hovering over diagnostic programs and a volt/ohm meter; mumbling unprintable incantations. All in all, living in a computerized house is a lot of fun, although I do mind being locked out and called an invalid file.

(Art Coughlin lives in New Jersey and works for the State Department. He has made his living for the past 16 years programming mainframes, minis and micros but still prefers his Apple IIgs.)

Miscellanea

System 6, System 6, where art thou System 6? If we had a 900 number for System 6 inquiries everyone here at the office would be vacationing in Hawaii by now. We get a lot of mail and a mega amount of phone calls every day on the subject. When our last issue and catalog went to press we thought that it was, literally, in the mail. It wasn't. We have to commend, not condemn the System 6 Development Team at Apple for trying to put out as perfect a product as is humanly possible. As I write this we are still not sure of the actual release date but rest assured that it **will** arrive. And when it does, you can be sure that we will be working overtime to get it into the hands of those of you who have ordered it.

Beagle telephone support is up and running again. Now that ***BeagleWorks,*** their first Macintosh package, is shipping, they have reallocated personnel to Apple II support. Thanks to Chuck Newby of Charlie's Appleseeds and Beverly Cadieux of Texas II for this late breaking tip.

Last month we reported an irritating bug in *HyperStudio 3.0* that had to do with the inability to save stacks to a disk with a volume name longer than 12 characters. No sooner than the issue hit the newsstand did Eric Mueller of RWP call to announce that the bug, as well as numerous others, had been squashed. And while they were busy swatting bugs, they also improved the font selection box (adding the ability to tab between controls), added key equivalents for almost every button, and made large-stack handling faster. And you didn't expect them to add all these improvements without adding a version number, did you? The *HyperStudio 3.1* upgrade is available at no charge if you want to send back your original *HyperStudio 3.0* disks. If you want to hold on to them for posterity, send a check for $10.00 and the fine folks at Roger Wagner Publishing will send you a new set. If you own version 2.1 and want to upgrade to version 3.1, the cost is $50.00. For further information contact them at 1050 Pioneer Way, Suite P, El Cajon, Calif. 92020, 619-442-0522, fax 619-442-0525.

An update to Desktop Enhancer is now available from Simplexity Software (1305 Chapman Avenue, Suite 302, Orange, Calif. 92668, 714-283-3957) This update, version 1.1, should take care of the $1104 errors some were experiencing.

Health professionals can now take a shortcut to finding software specifically tailored to meet their needs. Medical Software Products (MSP) announced a new directory of over 100 specialized products for physcians and health care professionals. Products listed contain descriptions, prices, system requirements and ordering information. Not only does MSP require that all items have a record of successful installation in the appropriate facilities, but they also offer a full 30-day guarantee of satisfaction. For a free *Medical Software Catalog* write to Medical Software Products, 591 W. Hamilton Avenue, Suite 205, Campbell, Calif. 95008, call 800-444-4570 or fax 408-370-3393.

The nominees for the 1991 Apple II Achievement Awards have been announced by Apple Computer's Matt Deatherage. This year's nominees were selected by a committee of Apple II industry professionals including representatives from Apple Computer, America Online, Genie's Apple II forums, Compuserve's MAUG Apple II forums, GS+ Magazine, A+/Incider, Nibble Magazine, and Resource Central. The winners will be announced at an online conference on America Online on April 3, 1992 at 10 p.m. (est).

Each and every nominee deserves credit. While it is certainly true that we don't have as much as the MS-DOS world to chose from, what we do have is exceptional. So to give credit where credit is due, I'd like to list all the nominees along with their addresses.

Best Freeware or Shareware Program: *Milestones 2000* (Dr. Ken Franklin, 1603 Northridge Ct, Clarksville, Tenn. 37042, $15.00 Reliefware), *Nifty List 3.3* (DAL Systems, Box 875, Cupertino, Calif. 95015-0875, $15.00 shareware), *ShrinkIt GS 1.04* (Andy Nicholas, 1180 Reed Avenue, Apt 12, Sunnyvale, Calif. 94086), *SuperView 2.2* (Chris McKinsey, 1006 Mathew Drive, O'Fallon, IL 62269, $10.00 shareware fee),and *UtilityWorks GS* (George Wilde, GRW Systems, 24402 Broadwell Ave. Harbor City, Calif 90710, $20.00 shareware fee).

Best Educational Program: *GeoQuiz* (PC Globe, 4700 South McClinock, Tempe, Ariz. 85282, 602-730-9000), *HyperCard IIgs* (Apple Computer, 20525 Mariani Avenue, Cupertino, Calif. 95014), *HyperStudio 3.0* (Roger Wagner Publishing, 1050 Pioneer Way Suite P, El Cajon, Calif. 92020, 619-442-0525), *McGee at the Fun Fair* (Lawrence Productions, distributed by Broderbund.)

Best 8-bit Application: *Delta Drawing Today* (Power Industries LP, 37 Walnut St, Wellesley Hills, MA 02181, 800-395-5009), *InWords* (Westcode Software 11835 Carmel Mountain Road, Suite 1304, San Diego, Calif. 92128, 619-679-9200), *ProTERM 3.0* (inSync Software, 3035 E. Topaz Circle Phoenix, Arizona 85028-4423), *PublishIt 4* (TimeWorks, 444 Lake Cook Road Deerfield, IL60015), and *Total Control* (JEM Software, 303-422-4766 or fax 303-422-4856.

Best 16-bit Application: *DreamGraphix* (DreamWorld Systems 1956 Broadway #B6 Iowa City, IA 52240, 319-354-7959), *GraphicWriter III version 1.1* (Seven Hills Software 2310 Oxford Road Tallahassee, FL 32304-3930, 904-575-0566), *HyperCard IIgs* (Apple Computer, (Apple Computer, 20525 Mariani Avenue, Cupertino, Calif. 95014), *HyperStudio 3.0* (Roger Wagner Publishing, 1050 Pioneer Way Suite P, El Cajon, Calif. 92020, 619-442-0525), *SuperConvert* (Seven Hills Software, 2310 Oxford Road, Tallahassee, Fla. 32304, 903-575-0566).

Best Innovation: *DreamGraphix* (DreamWorld Systems 1956 Broadway #B6 Iowa City, IA 52240, 319-354-7959), *HyperCard IIgs* (Apple Computer, (Apple Computer, 20525 Mariani Avenue, Cupertino, Calif. 95014), *InWords* and *Pointless* (Westcode Software 11835 Carmel Mountain Road, Suite 1304, San Diego, Calif. 92128, 619-679-9200), *System Software 6.0* (Apple Computer, 20525 Mariani Avenue, Cupertino, Calif. 95014).

Best Multimedia Achievement: *HyperBole* (Resource Central, Box 11250, Overland Park, KS 66211, 913-469-6502), *HyperCard IIgs* (Apple Computer, 20525 Mariani Avenue, Cupertino, Calif. 95014), *HyperStudio 3.0* (Roger Wagner Publishing, 1050 Pioneer Way Suite P, El Cajon, Calif. 92020, 619-442-0525), *Media Control Tool* (Apple Computer, 20525 Mariani Avenue, Cupertino, Calif. 95014), *ScriptCentral* (Resource Central, Box 11250, Overland Park, KS 66211, 913-469-6502).

Best Utility: *Finder 6.0* (Apple Computer, 20525 Mariani Avenue, Cupertino, Calif. 95014), *Prosel 16* (Glen Bredon 521 State Road,

Princeton, NJ 08540), *Salvation Series* (Vitesse, Inc 13909 Amar Road Suite 2A, La Puenta, Calif. 91746), *ShrinkIt GS* (Andy Nicholas, 1180 Reed Avenue., Apt 12, Sunnyvale, Calif. 94086), *SuperConvert* (Seven Hills Software, 2310 Oxford Road, Tallahassee, Fla. 32304, 903-575-0566).

Outstanding Developer Aid: *GSBug 1.6* (Apple Computer, 20525 Mariani Avenue, Cupertino, Calif. 95014), *GNO/ME* (Procyon, Inc. 1005 N. Kingshighway, Suite 309, Cape Girardeau, MO 63701, 314-334-7078), *Nifty List 3.3* (DAL Systems, P.O. Box 875, Cupertino, Calif. 95015-0875, $15.00 shareware), *ORCA/M 2.0* and *Talking Tools* (Byte Works, Inc. 4700 Irving Blvd. NW, Suite 207, Albequerque, NM 87114),

Best Apple II publication: *A2-Central* (Resource Central, Box 11250, Overland Park, KS 66211, 913-469-6502), *A+/Incider* (IDG Communications, Box 50358, Boulder, CO 80321), *GS+ Magazine* (EGO Systems, Box 15366 Chattanooga, Tenn. 37415-0366), *Nibble Magazine* (MindCraft Publishing, 52 Domino Drive, Concord, MA 01742- 9906), *Script-Central* (Resource Central, Box 11250, Overland Park, KS 66211, 913-469-6502).

Best Apple II Online Service: America Online, CompuServe, Delphi, GEnie, The Internet (including all Apple II newsgroups and mailing lists).

Software of the Year: *DreamGraphix* (DreamWorld Systems), *HyperCard IIgs* (Apple Computer), *HyperStudio 3.0* (Roger Wagner Publishing), *Pointless* (Westcode Software), *ProTERM 3.0* (inSync Software), *System Software 6.0* (Apple Computer).

Just let me mention two new releases: *GNO/ME* (Procyon, see address above), the multitaking environment for the Apple IIgs and *Accudraw Floor Plan* for Apple IIe, c, c plus, Laser, or Mac LC with IIe emulation card. Contact Kitchen Sink Software, 903 Knebworth Ct., Westerville, OH 43081, 614-891-2111. More to come.

Ask (or tell) Uncle

The celebration continues

I met my first Apple, a 48k Apple II Plus in 1981. I was in fifth grade at the time. After programming on my school's computer for a couple of years, I convinced my parents that they needed a computer for their business. They're on their second computer now, an accelerated Apple IIe. It handles all of the accounting for two family businesses and it processes continuous inventory and sales data for a medium-volume nutritional products business. After several years of prodding, I finally convinced them to try AppleWorks; now the typewriter only gets fired up to fill in forms.

I purchased an Apple IIgs when I started college at Iowa State University (Go Clones!). There isn't a day that goes by that I don't use the machine. I'm studying electrial engineering, a choice made in a large part because of my association with the open architecture of the Apple II. After three summers working for a very Blue company, my dedication to the Apple is stronger than ever.

At one point I was contemplating purchasing a Macintosh and saying goodbye to the Apple II community. But after purchasing Micol Advanced Basic, an HP DeskJet 500, Independence printer drivers, and another meg of memory, I'll take the flexibility of my Apple IIgs over the Mac anyday. Thanks for the product reviews and info!

Your article on games for the Apple II line (*"Imagination rules the world"*, December 1991) inspired me to dust off the programming mitts and see if I could learn to program in the Desktop environment. After a lot of late nights of cursing and celebrating, *The Dragon* was the result. I'd like to release the game as shareware but I don't have access to GEnie. *(The Dragon is a very nice Shanghai clone. I probably spent just a little too long making sure that it worked!—edr)* If you could put the game into circulation, it would be deeply appreciated. I would also be happy to send a copy to anyone who would like it; just send a blank disk and $10.00. I promise a one day turn around (unless I have an accounting test the next day!).

The main program (TheDragon) can be installed anywhere but it requires the font file DRAGON to be in the fonts folder. This is somewhat awkward but seems to work. The file DRAGON.DAT must reside in the same directory as the main program. Thanks to Stephen Hadinger's Fontasm for the graphics tool!

Curt Clifton
P.O. Box 306
110 N. West Street
Earlville, Iowa 52041

If you would like to join our party, write us. Tell us tell us in your own words what it is about this computer that makes you so loyal to it. We'll publish a letter each month and celebrate all year long.—edr

CD-ROM access

I was intrigued to read in the February 1992 issue of ***A2-Central*** that you have been able to read IBM CD-ROM disks on an Apple IIgs. I understand that you were using graphic files, but the idea of accessing the extensive world of IBM CD-ROM platters interests a number of members in our user group. (As you know, there are only one or two CD-ROM platters available for the Apple II.)

Can you supply a little more detail on your procedure? I would think that most people are interested in the information access potential of CD-ROM, and of course, that means IBM.

A couple of questions:

1. Are there other CD-ROM drives usable by System 6 or are we limited to the expensive Apple drive?

2. Are text and graphics both discernible?

3. What special hardware/software is needed to read IBM platters?

Any and all suggestions and direction would be greatly appreciated. Any cost ideas?

Stephen P. Hartz D.V.M.
Crestline, Ohio

You want an Apple II success story? I operate a $500M+ gross volume business selling (mainly) catalysts from my home using an Apple IIc and an Apple IIgs with three 3.5 drives. We have been in business since 1983 and will probably begin manufacturing in 1992.

While I see the need for a hard drive, I really need to have access to some of the information on CD's for business applications. Is there any way to use the PC Transporter I have in the Apple IIgs to hook up to a CD? Can I use something like HyperCard to get this information? Or do I have to get a machine running MS-DOS?

Nancy Irwin
Cleveland, Ohio

The major hurdle in using IBM CD-ROMs on the Apple IIgs is the data format of the file you want to import. By "data format" I'm asking questions like "Is the CD-ROM-based information that you want to access in an ASCII text file? In a Lotus 1-2-3 file? In a WordPerfect file? In a dBASE IV file? In a unique, encrypted file designed by the CD-ROM's publisher?

If the CD-ROM you want to access uses a generic ASCII text format that you can read from AppleWorks or AppleWorksGS, then the amount of work necessary to access the data (assuming you've installed the CD-ROM and the software on the IIgs) is:

1) Insert the CD-ROM platter
2) Read the file.

Period. End of story. All MS-DOS CD-ROMs we've seen use the High Sierra directory protocol. The way GS/OS works, the High Sierra File System Translator, which comes with the GS/OS system software, will read the directory and the files on a CD-ROM just like it was any other file system installed under GS/OS.

We've only run into two problems: SCSI cards that don't seem to work with file systems that aren't a lot like ProDOS (High Sierra CD

ROMs use a "block size" that's a lot bigger than than the 512-byte blocks that ProDOS uses) and programs that do stupid things with GS/OS (such as assume all filenames will be valid ProDOS filenames, when GS/OS actually uses much broader naming conventions).

What these two problems mean is that the only combination of equipment we've been able to read High Sierra CD-ROMs with so far is an Apple DMA SCSI card connected to an AppleCD SC. We've also heard that a company called Peripheral Land makes a CD-ROM drive that will work with an Apple SCSI card. If anyone has another combination that works, we'd like to hear about it. We've gotten lots of people asking questions about cheaper drives, but very few answers.

We have not have success using the RamFast SCSI card trying to read High Sierra CD-ROMs. We believe the problem is the odd block size High Sierra uses and we've let C.V. Tech know.

According to Applied Engineering, the PC Transporter is not prepared to access MS-DOS CD-ROM drives attached to the IIgs (the disk drivers were not written to handle that function).

But if you spend the money and get the Apple equipment, accessing files on the CD-ROM is transparent. Text (".TXT" or ".DOC" files) can be opened with ***AppleWorksGS,*** *".GIF" graphics files can be opened and converted with* ***SuperConvert,*** *and ".PCX" files can be opened with* ***The Graphics Exchange.*** *We even took an* ***Autodesk Animator*** *".FLI" animation file, split out the individual frames on our hard disk with FLIAway (on the* ***SuperConvert*** *disk), imported and converted them with* ***SuperConvert,*** *then built a* ***HyperStudio*** *animation out of the frames. It's that easy, and there really isn't anything to elaborate.*

On the other hand, if you have a file that's in a proprietary format, there is much elaboration to do. Basically, it breaks down to determining the format of the file, then writing a translator to convert it to a form usable on the IIgs. There isn't any way to give a "generic" description of that; it's like trying to explain how to solve a jigsaw puzzle in simpler terms than "assemble the pieces that fit together properly".

The only clue we can give is that ***HyperCard IIgs*** *is well-suited to doing some of the conversion since its "READ FILE" command can read any type of file as raw data. Using this technique, you can load a chunk of a file and "parse" it in much the same way as you would BLOAD a chunk of an AppleWorks file in BASIC and PEEK your way through it (see "Reading AppleWorks data bases", March 1987).*

There are some weaknesses in ***HyperCard's*** *file-handling capabilities if you have to deal with "random access" files. There is no way to "position" yourself in a file other than by reading and discarding bytes, which is much slower than using random access methods that go directly to the data needed without first reading everything that comes before it. And* ***HyperCard*** *replaces "null" bytes with spaces, destroying their original value as data; this prevents you from writing a generic "reader" for standard database formats. But an enterprising individual (hint, hint) could probably write Xcmds to surmount these minor obstacles and turn* ***HyperCard*** *into a more satisfactory front end.*

But the exact technique for reading files will vary for each file format, and files that are compressed (to save space) or encrypted (to prevent theft of intellectual property) are going to be particularly difficult to decipher.

On the other hand, the structure of common file formats, like those of dBASE data files or WordPerfect word processor files or Lotus 1-2-3 spreadsheet files may be defined somewhere. In this case working out a translator would mostly be a matter of "plug and chug" programming. For such broadly accepted formats, a standalone application that could read the format might be useful with a whole series of CD-ROMs. (The ability to write in the same format wouldn't be needed in this context, since the original file is on a read-only medium.)

Unless there's something we've specifically overlooked, this is about as far as we can take the example ourselves. In the meantime, if users find configurations that work, that information would be useful to others (including Apple II developers debating whether to work on CD-ROM projects). Let us hear about it!—DJD

A2-Central™

[redacted]

Publisher: **Tom Weishaar** Editor: **Ellen Rosenberg**

with help from:
Denise Cameron **Dennis Doms** **Sally Dwyer**
Dean Esmay **Richard Ginter** **Jeff Neuer**
Jean Weishaar

A2-Central,—titled ***Open-Apple*** through January, 1989—has been published monthly since January 1985. World-wide prices (in U.S. dollars; airmail delivery included at no additional charge): $34 for 1 year; $60 for 2 years; $84 for 3 years. All back issues are currently available for $2 each; bound, indexed editions of our first six volumes are $14.95 each. Volumes end with the January issue; an index for the prior volume is included with the February issue.

The full text of each issue of ***A2-Central*** is available on 3.5 disks, along with a selection of the best new public domain and shareware files and programs, for $90 a year (newsletter and disk combined). Single disks are $10.

Please send all correspondence to:

A2-Central
P.O. Box 11250
Overland Park, Kansas 66207 U.S.A.

A2-Central is sold in an unprotected format for your convenience. You are encouraged to make back-up archival copies or easy-to-read enlarged copies for your own use without charge. You may also copy ***A2-Central*** for distribution to others. The distribution fee is 20 cents per page per copy distributed.

WARRANTY AND LIMITATION OF LIABILITY. We warrant that most of the information in ***A2-Central*** is useful and correct, although drivel and mistakes are included from time to time, usually unintentionally. Unsatisfied subscribers may cancel their subscription at any time and receive a full refund of their last subscription payment. The unfilled portion of any paid subscription will be refunded even to satisfied subscribers upon request. OUR LIABILITY FOR ERRORS AND OMISSIONS IS LIMITED TO THIS PUBLICATION'S PURCHASE PRICE. In no case shall our company or our contributors be liable for any incidental or consequential damages, nor for ANY damages in excess of the fees paid by a subscriber.

ISSN 0885-4017 **GEnie mail: A2-CENTRAL**

Apple II support

The Alliance International, Inc. came about as a result of discussions between a group of Apple II developers, users and Apple engineers on *America Online's* "Across The Boards" area. One of the developers/users involved in those discussions, John Majka, of Louisville, Ken., decided to form a "support group" for the Apple II based on the premise that since Apple Computer, Inc. apparently is not going to support Apple II owners, Apple II owners should get together and support each other.

In late 1991, *All* was incorporated in the state of Kentucky. It is currently a "for-profit" corporation because a "non-profit" company is somewhat more complicated to set up and there are complex state and federal laws about tracking donations, costs, reporting, etc. So it is simpler and cheaper to go "for-profit", at least for a while.

The mission of The Alliance is to promote and advance the use of the Apple II computer in home, business, education, and other markets.

We intend to accomplish that mission by:

1. Notifying Apple II users through national media (newspapers, radio, TV) that they can get support for their computer from The Alliance.

2. Providing an "800" support number.

3. Offering a subscription to a quarterly publication which lists available Apple II software, hardware and services and where to purchase it or find it.

4. Encouraging existing developers of Apple II software to continue development.

5. Encouraging developers of software for other computers (*Borland, Aston-Tate, Lotus,* etc.) for home and small businesses to develop software for the Apple II.

6. Talking to, lobbying, pressuring Apple Computer, Inc. to devote more resources to the Apple II line.

In order to accomplish these goals we are asking individuals and Users Groups to become members of the Alliance. The dues for individuals are $20 and for Users Groups, $50.

Your membership will tell us that **you** want the Apple II to live and grow. It will tells us, and the developers that you want more software and hardware. It will tell us if we are right or if Apple Computer, Inc. is right, that Apple II users don't care, and that we are just whistling in the grave yard of the Apple II. It will let us know if we are wasting our time, efforts, and money or not.

If you decide to support us in supporting the Apple II, please send your $20 or $50 check and any ideas and suggestions you may have to:

The Alliance International Inc.
P.O. Box 20756
Louisville, KY 40250.